Combinational Logic

Prof. Usagi


DON'T
WORRY
BE
HAPPY

Recap: Logic Design?

Logic design

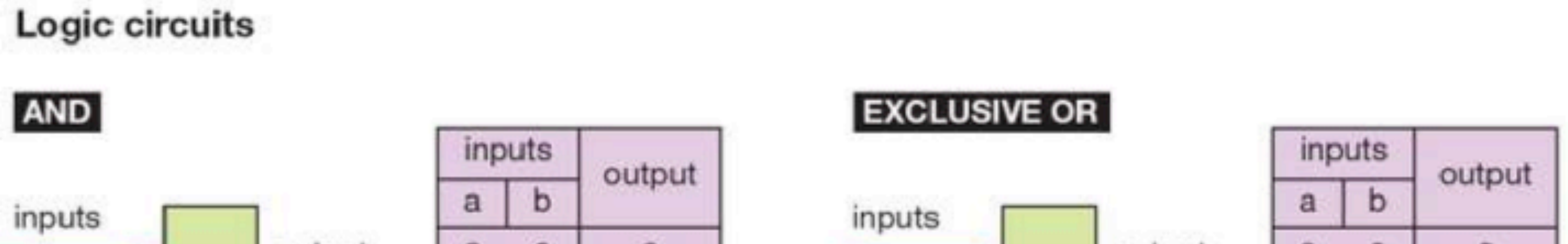
https://www.britannica.com/technology/logic-design

COMPUTER TECHNOLOGY

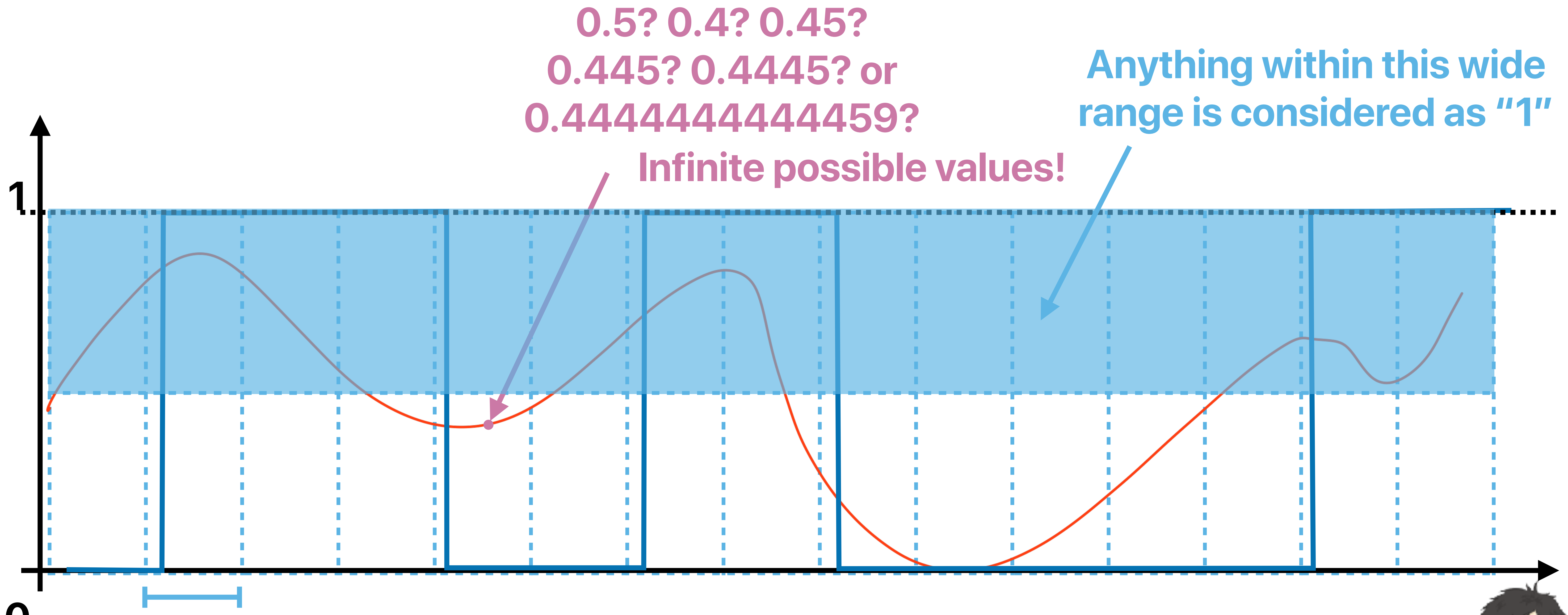
WRITTEN BY: The Editors of Encyclopaedia Britannica

See Article History

Logic design, Basic organization of the circuitry of a digital computer. All digital computers are based on a two-valued logic system—1/0, on/off, yes/no (*see* binary code). Computers perform calculations using components called logic gates, which are made up of integrated circuits that receive an input signal, process it, and change it into an output signal. The components of the gates pass or block a clock pulse as it travels through them, and the output bits of the gates control other gates or output the result. There are three basic kinds of logic gates, called “and,” “or,” and “not.” By connecting logic gates together, a device can be constructed that can perform basic arithmetic functions.



Recap: Analog v.s. digital signals



Recap: Why are digital computers more popular now?

- Please identify how many of the following statements explains why digital computers are now more popular than analog computers.
  - ✔ The cost of building systems with the same functionality is lower by using digital computers.
  - ✘ Digital computers can express more values than analog computers.
  - ✔ Digital signals are less fragile to noise and defective/low-quality components.
  - ✔ Digital data are easier to store.

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Recap: The basic idea of a number system

- Each position represents a quantity; symbol in position means how many of that quantity
  - Decimal (base 10)
    - Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    - More than 9: next position
    - Each position is incremented by power of 10
  - Binary (base 2)
    - Two symbols: 0, 1
    - More than 1: next position
    - Each position is incremented by power of 2

$$10^2 \times \mathbf{3} + 10^1 \times \mathbf{2} + 10^0 \times \mathbf{1} = 300 + 20 + 1 = 321$$

$$2^3 \times \mathbf{1} + 2^2 \times \mathbf{0} + 2^1 \times \mathbf{0} + 2^0 \times \mathbf{1} = 1 \times 2^3 + 1 \times 2^0 = 1 \times 8 + 1 \times 1 = 9$$

Outline

- Two types of logics
- The **theory** behind combinational logics
- The building blocks of combinational logics

Types of circuits

Combinational v.s. sequential logic

- Combinational logic
  - The output is a pure function of its current inputs
  - The output doesn’t change regardless how many times the logic is triggered — Idempotent
- Sequential logic
  - The output depends on current inputs, previous inputs, their history

Poll close in 1:30

When to use combinational logic?

- How many of the following can we simply use combinational logics to accomplish?
  ① Counters
  ② Adders
  ③ Memory cells
  ④ Decimal to 7-segment LED-decoders
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

When to use combinational logic?

- How many of the following can we simply use combinational logics to accomplish?
  ① Counters — You need the previous input
  ② Adders ✓
  ③ Memory cells — You need to keep the current state
  ④ Decimal to 7-segment LED-decoders ✓
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Theory behind each

- A **Combinational logic** is the implementation of a **Boolean Algebra** function with only Boolean Variables as their inputs
- A **Sequential logic** is the implementation of a **Finite-State Machine**

Boolean Algebra

Boolean algebra (disambiguation)

- Boolean algebra — George Boole, 1815—1864
  - Introduced binary variables
  - Introduced the three fundamental logic operations: AND, OR, and NOT
  - Extended to abstract algebra with set operations: intersect, union, complement
- Switching algebra — Claude Shannon, 1916—2001
  - Wrote his thesis demonstrating that electrical applications of **Boolean algebra** could construct any logical numerical relationship
  - Disposal of the abstract mathematical apparatus, casting **switching algebra** as the **two-element Boolean algebra**.
  - We now use switching algebra and boolean algebra interchangeably in EE, but not doing that if you're interacting with a mathematician.

Basic Boolean Algebra Concepts

- {0, 1}: The only two possible values in inputs/outputs
- Basic operators
  - AND (•) — a • b
    - returns 1 only if both a **and** b are 1s
    - otherwise returns 0
  - OR (+) — a + b
    - returns 1 if a **or** b is 1
    - returns 0 if none of them are 1s
  - NOT (‘) — a’
    - returns 0 if a is 1
    - returns 1 if a is 0

Truth tables

- A table sets out the functional values of logical expressions on each of their functional arguments, that is, for each combination of values taken by their logical variables

AND

Input		Output
A	B	
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

OR

Input		Output
A	B	
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

NOT

Input	Output
A	
0	1
0	1
1	0
1	0

Poll close in 1:30

Let’s practice!

- X, Y are two Boolean variables. Consider the following function:
  X • Y + X
  How many of the following the input values of X and Y can lead to an output of 1
  ① X = 0, Y = 0
  ② X = 0, Y = 1
  ③ X = 1, Y = 0
  ④ X = 1, Y = 1
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Let's practice!

- X, Y are two Boolean variables. Consider the following function:
  $X \cdot Y' + X$
  How many of the following the input values of X and Y can lead to an output of 1

  ① X = 0, Y = 0
  ② X = 0, Y = 1
  ③ X = 1, Y = 0 ✓
  ④ X = 1, Y = 1 ✓

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Input					
X	Y	Y′	XY′	XY′ + X	Output
0	0	1	0	0	**0**
0	1	0	0	0	**0**
1	0	1	1	1	**1**
1	1	0	0	1	**1**

Derived Boolean operators

- NAND — $(a \cdot b)'$
- NOR — $(a + b)'$
- XOR — $(a + b) \cdot (a' + b')$ or $ab' + a'b$
- XNOR — $(a + b') \cdot (a' + b)$ or $ab + a'b'$

NAND

Input		Output
A	B	
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

NOR

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

XOR

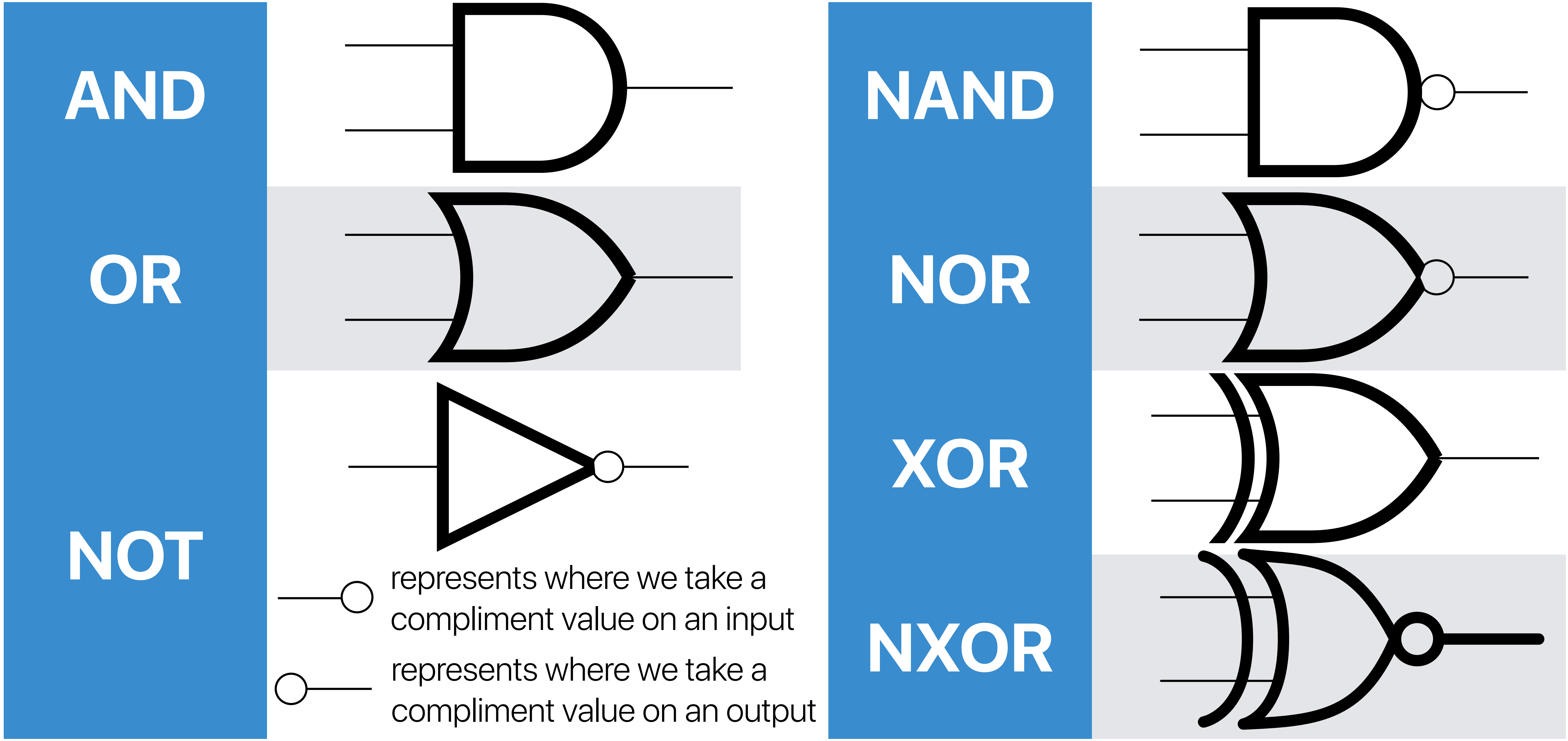
Input		Output
A	B	
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

XNOR

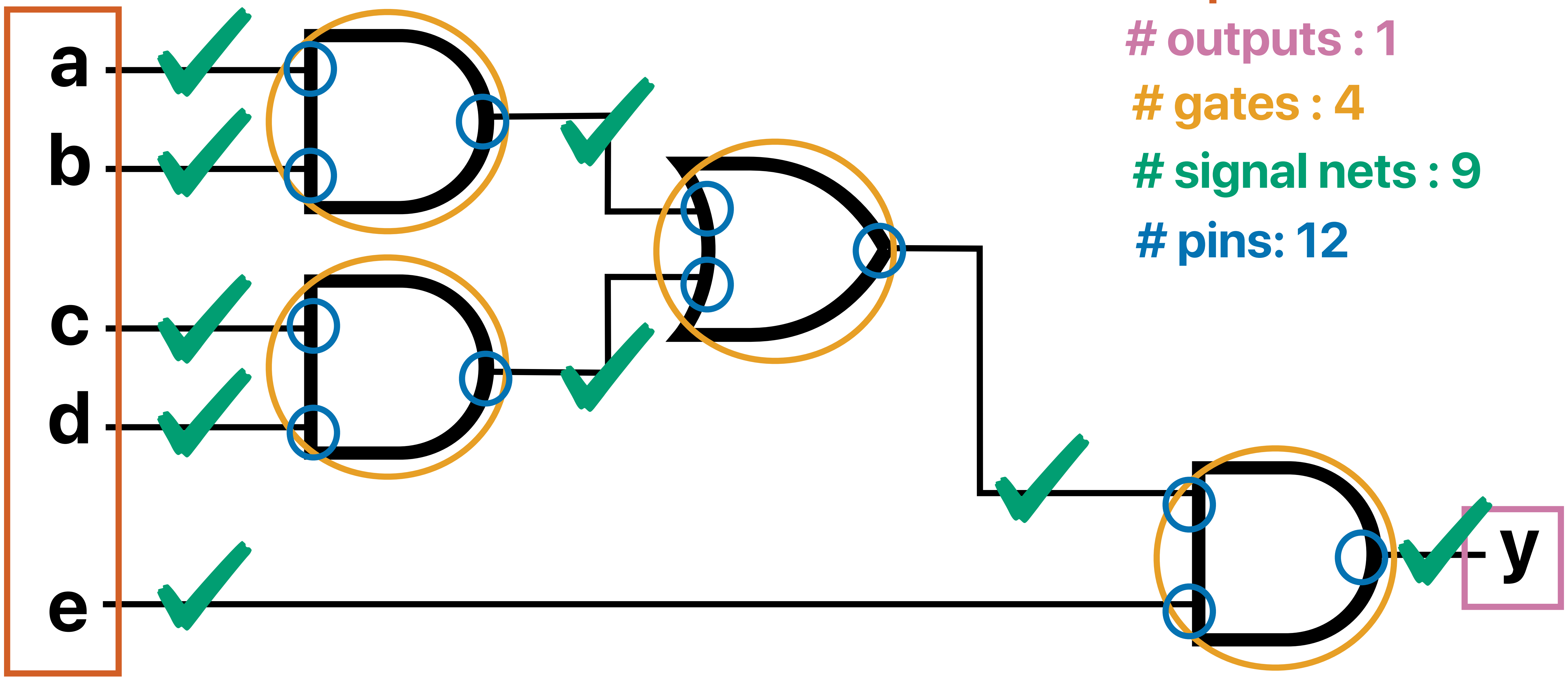
Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Express Boolean Operators/ Functions in Circuit “Gates”

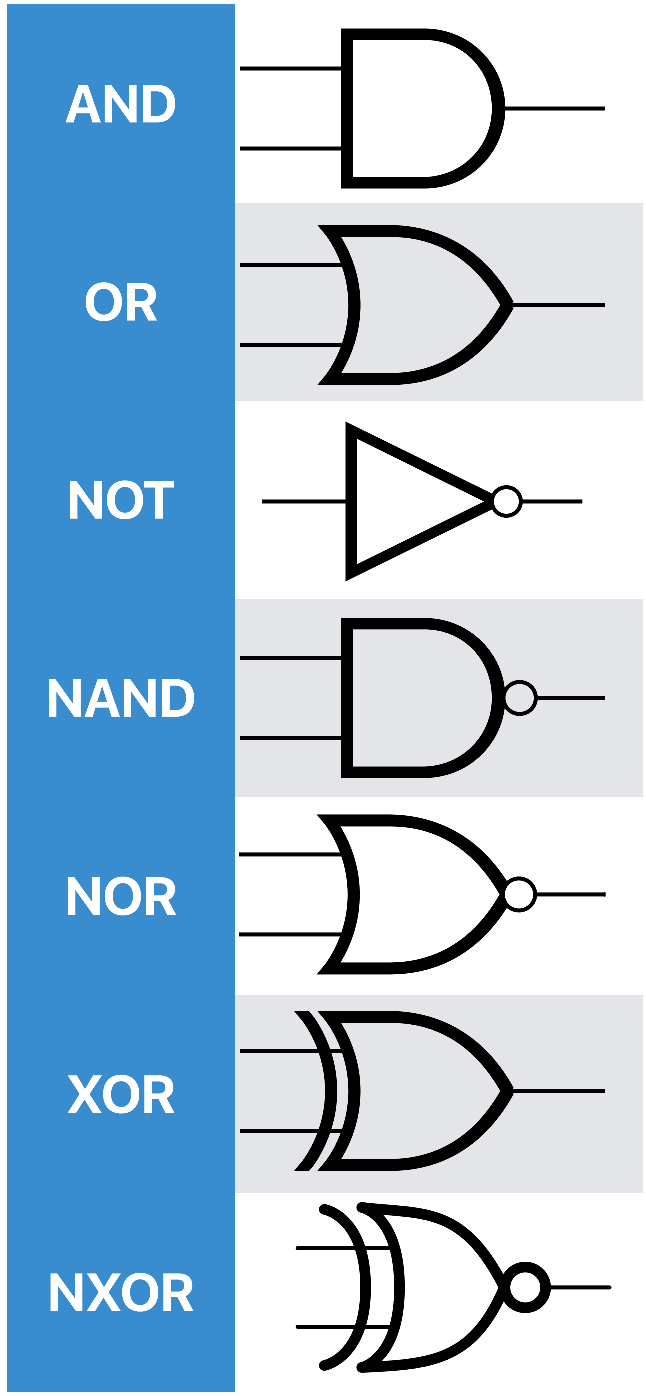
Boolean operators their circuit "gate" symbols



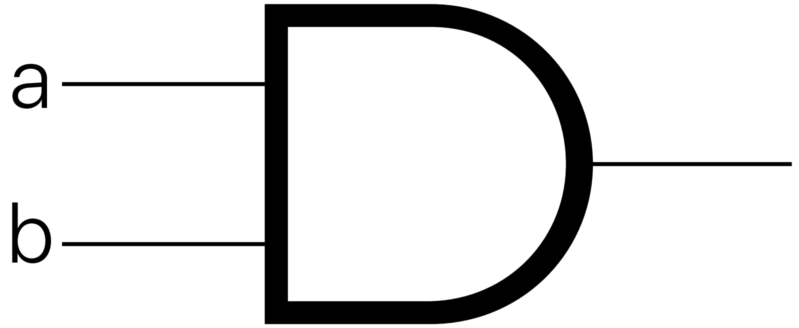
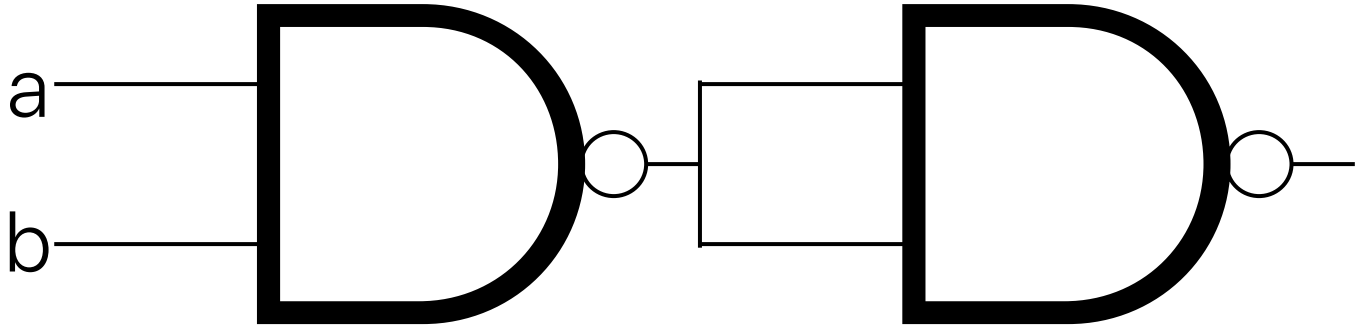
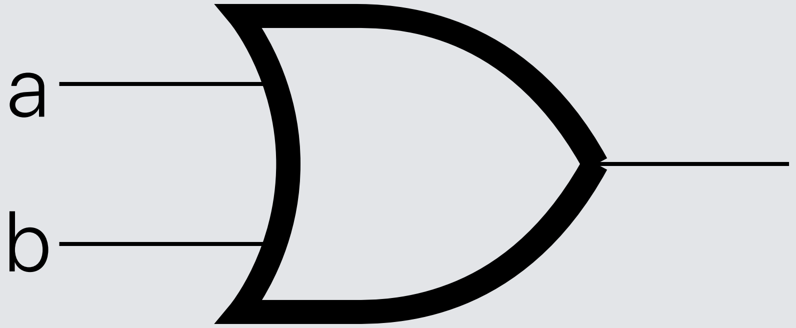
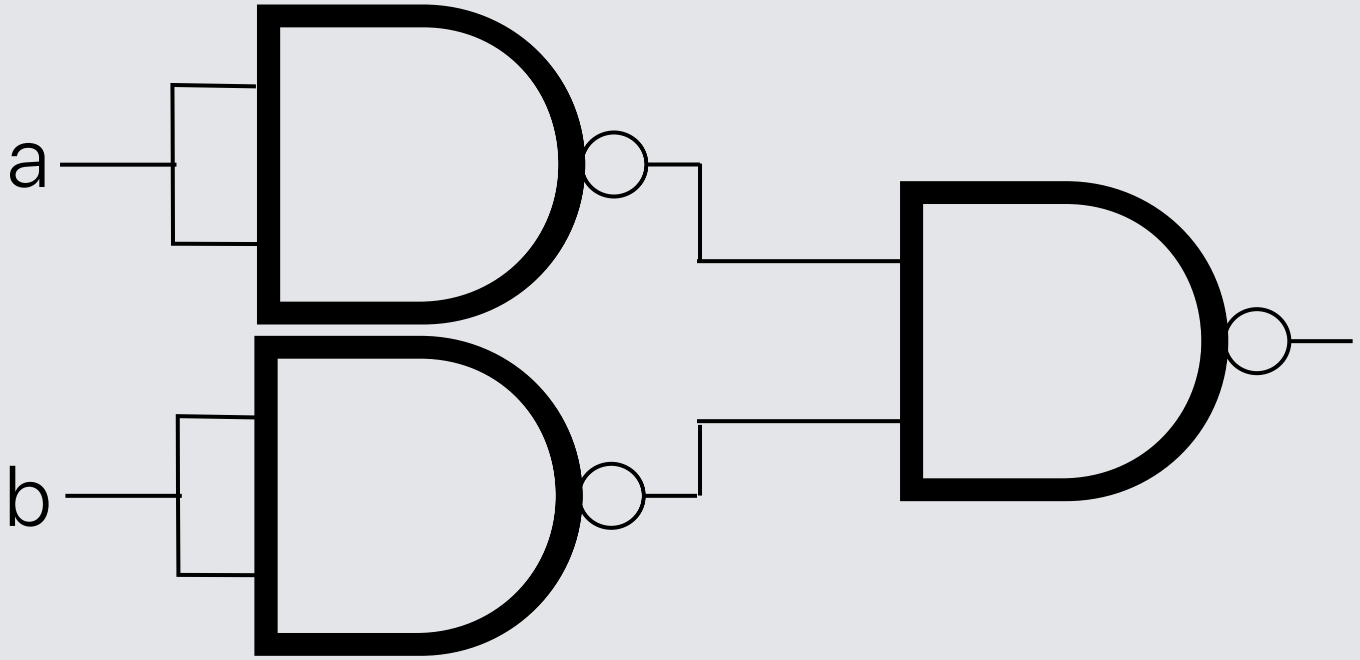
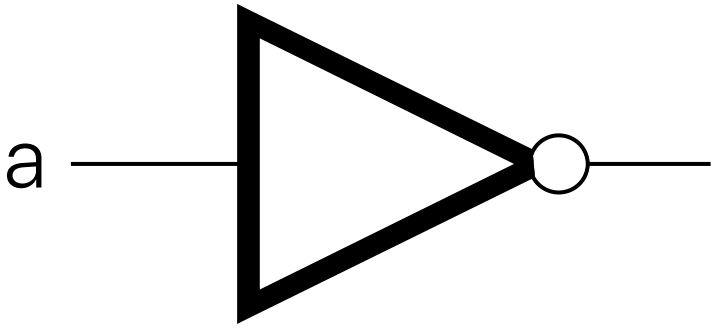
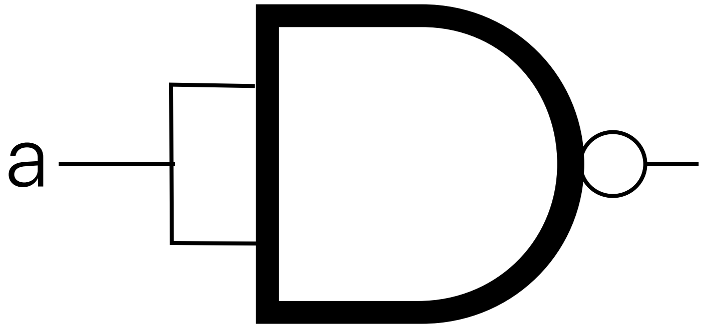
How to express y = e(ab+cd)



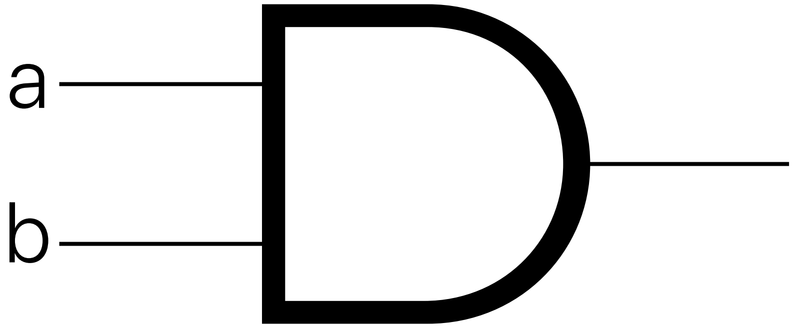
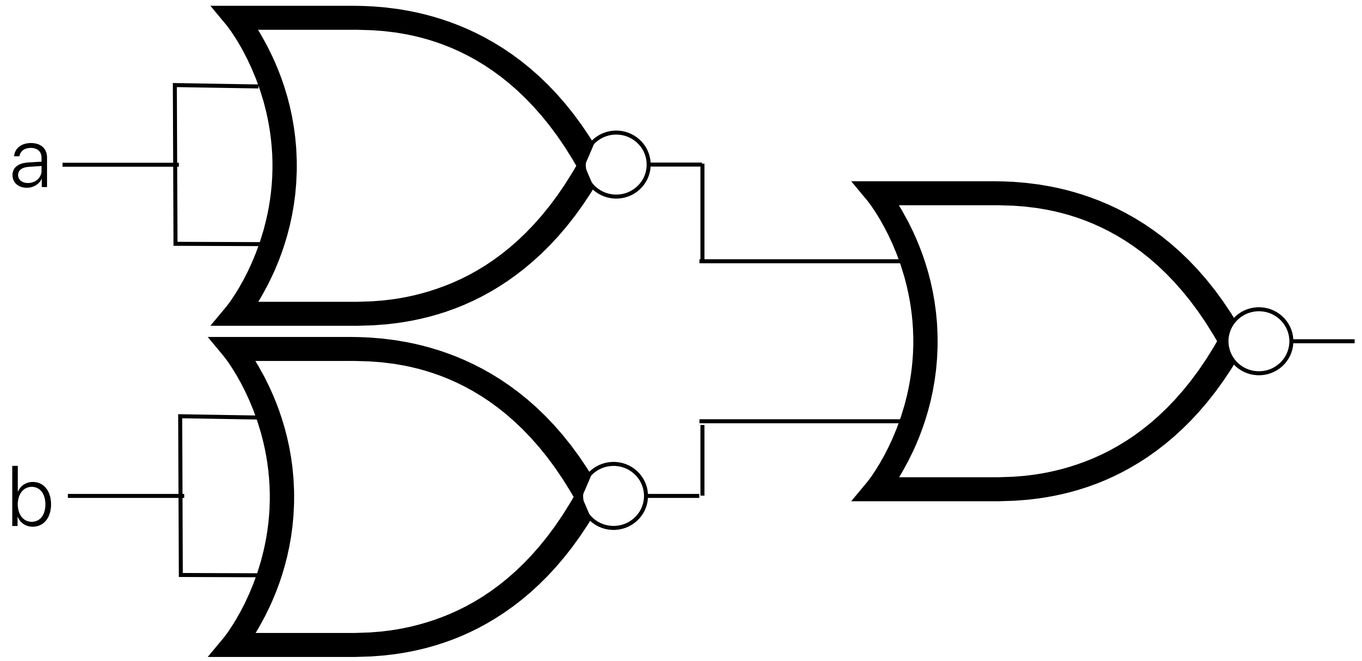
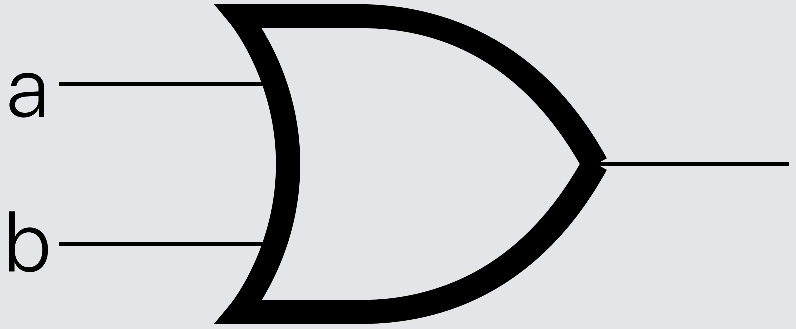
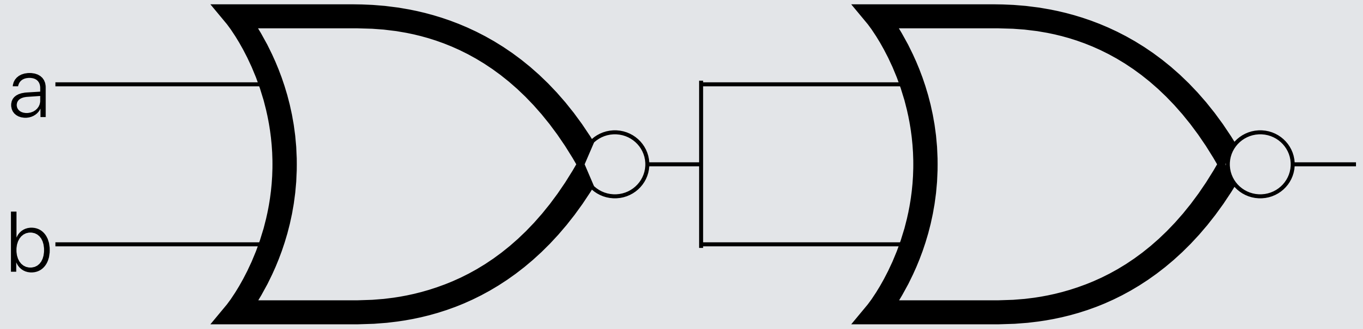
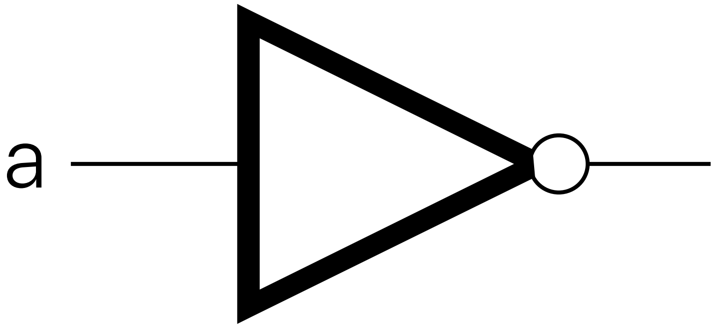
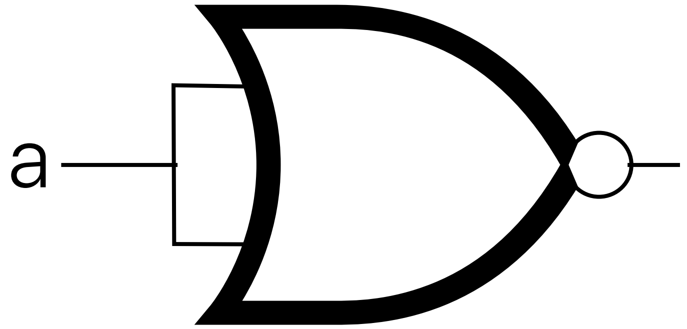
inputs : 5
outputs : 1
gates : 4
signal nets : 9
pins: 12



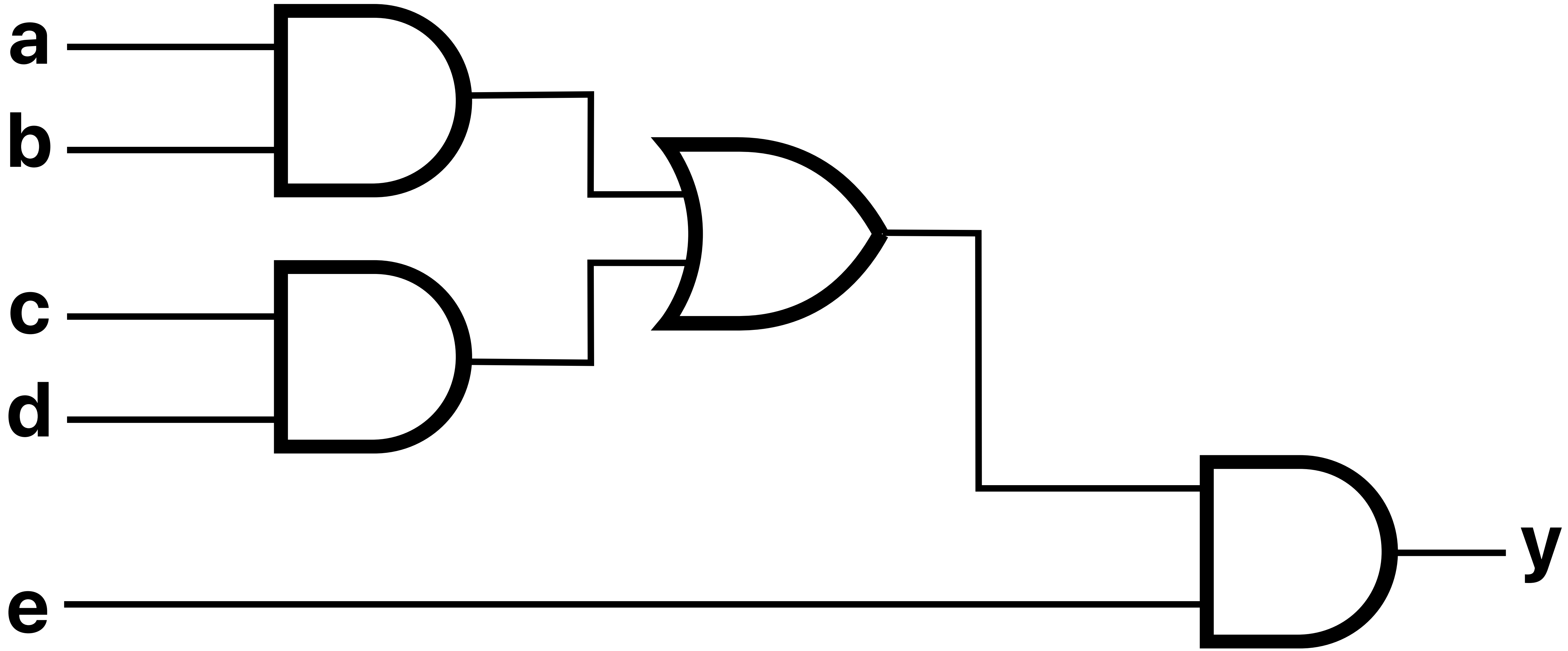
We can make everything NAND!

	Original	NAND
AND		
OR		
NOT		

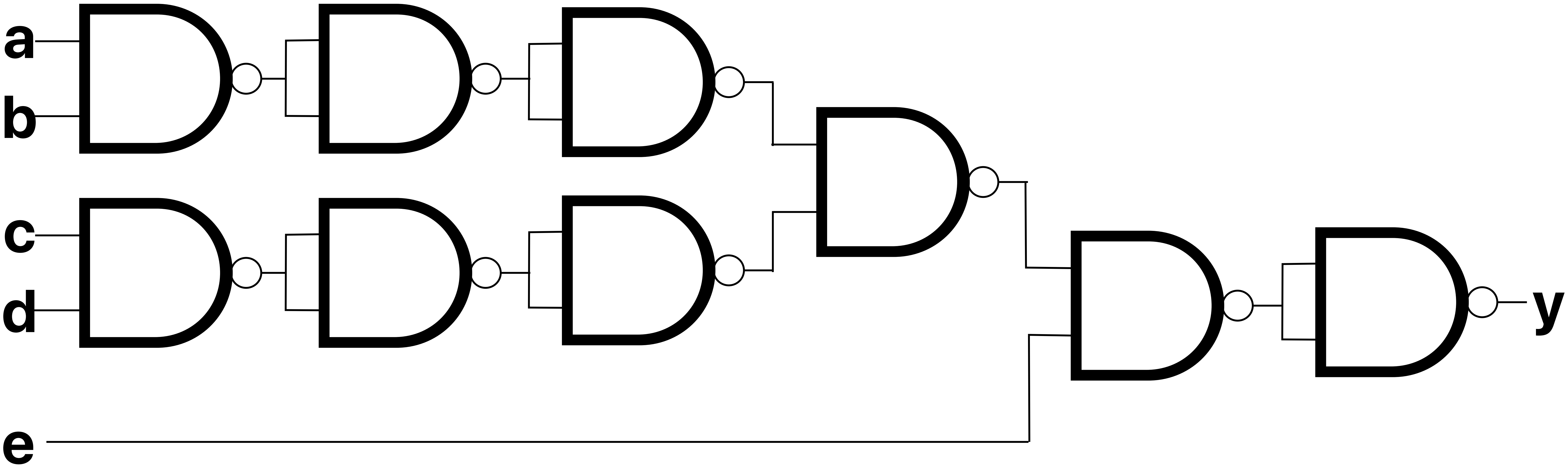
We can also make everything NOR!

	Original	NAND
AND		
OR		
NOT		

How to express y = e(ab+cd)



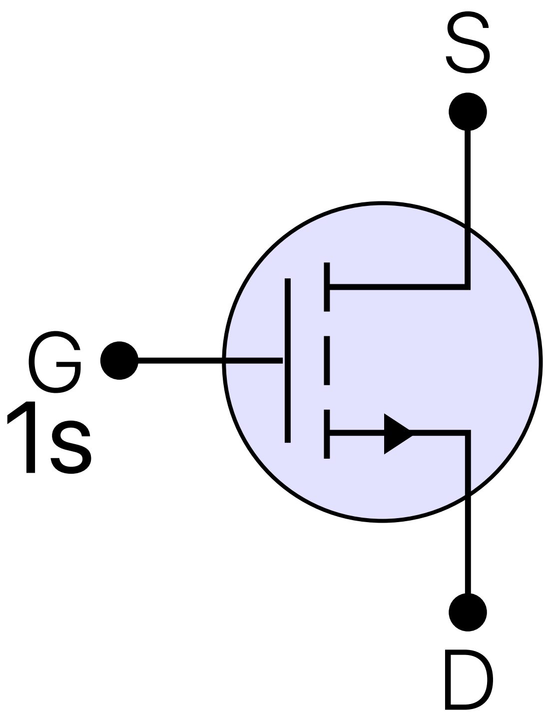
How to express y = e(ab+cd)



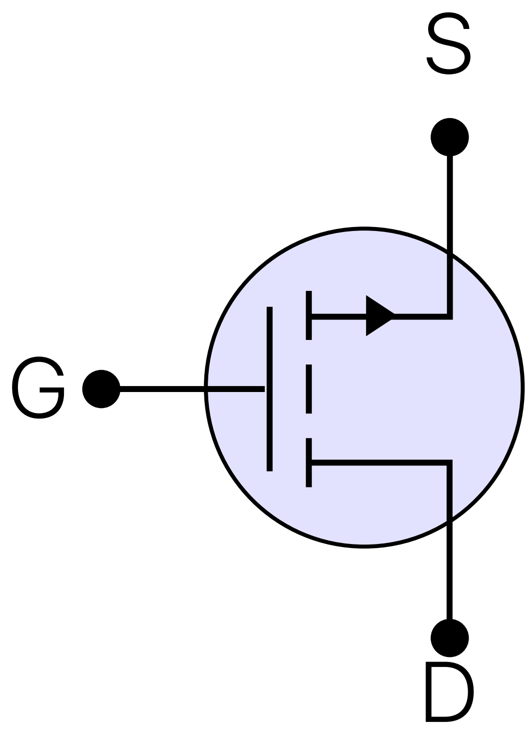
How gates are implemented?

Two type of CMOSs

- nMOS
  - Turns on when G = 1
  - When it's on, passes 0s, but not 1s
  - Connect S to ground (0)
  - Pulldown network

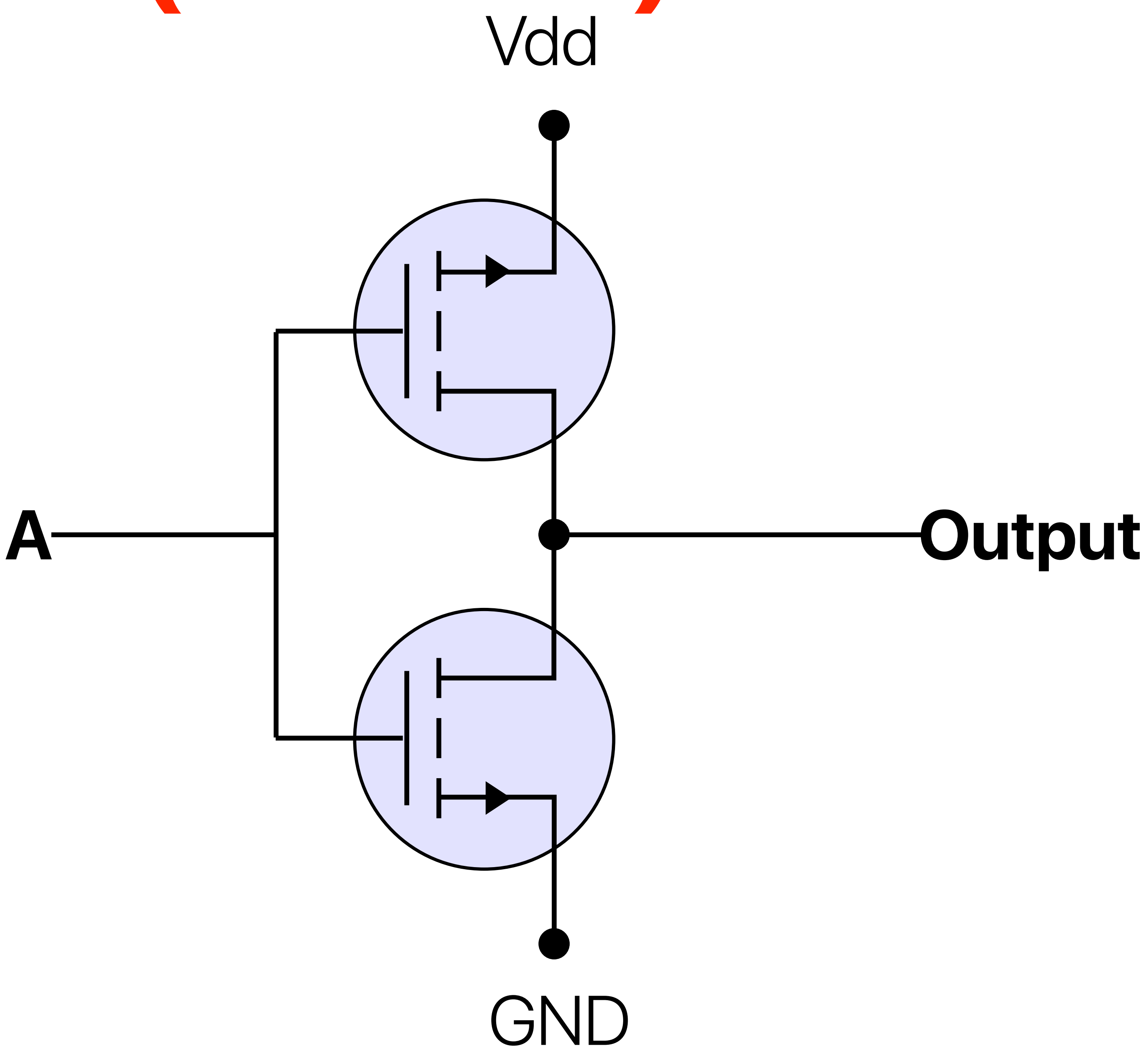


- pMOS
  - Turns on when G = 0
  - When it's on, passes 1s, but not 0s
  - Connect S to Vdd (1)
  - Pullup network

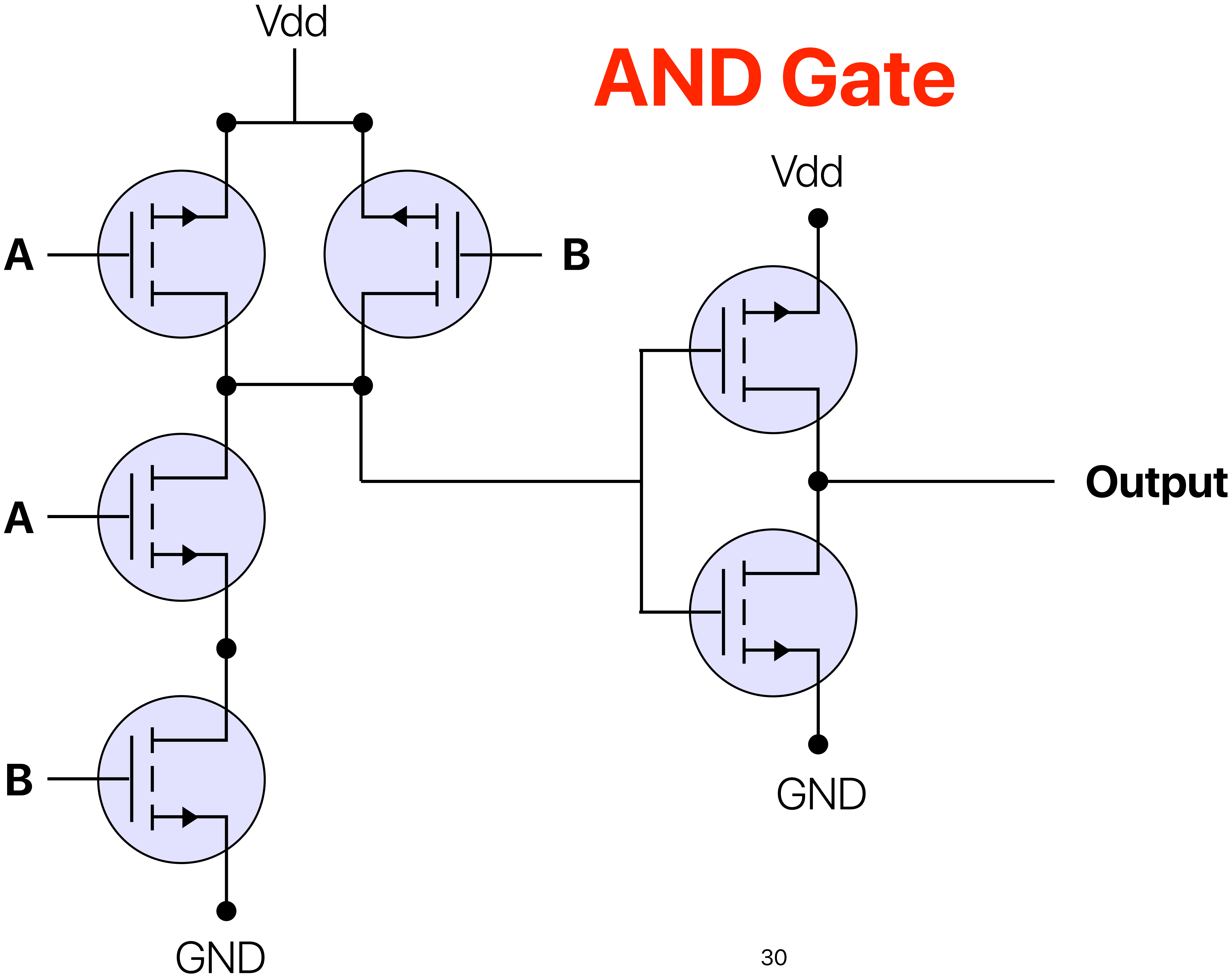


NOT Gate (Inverter)

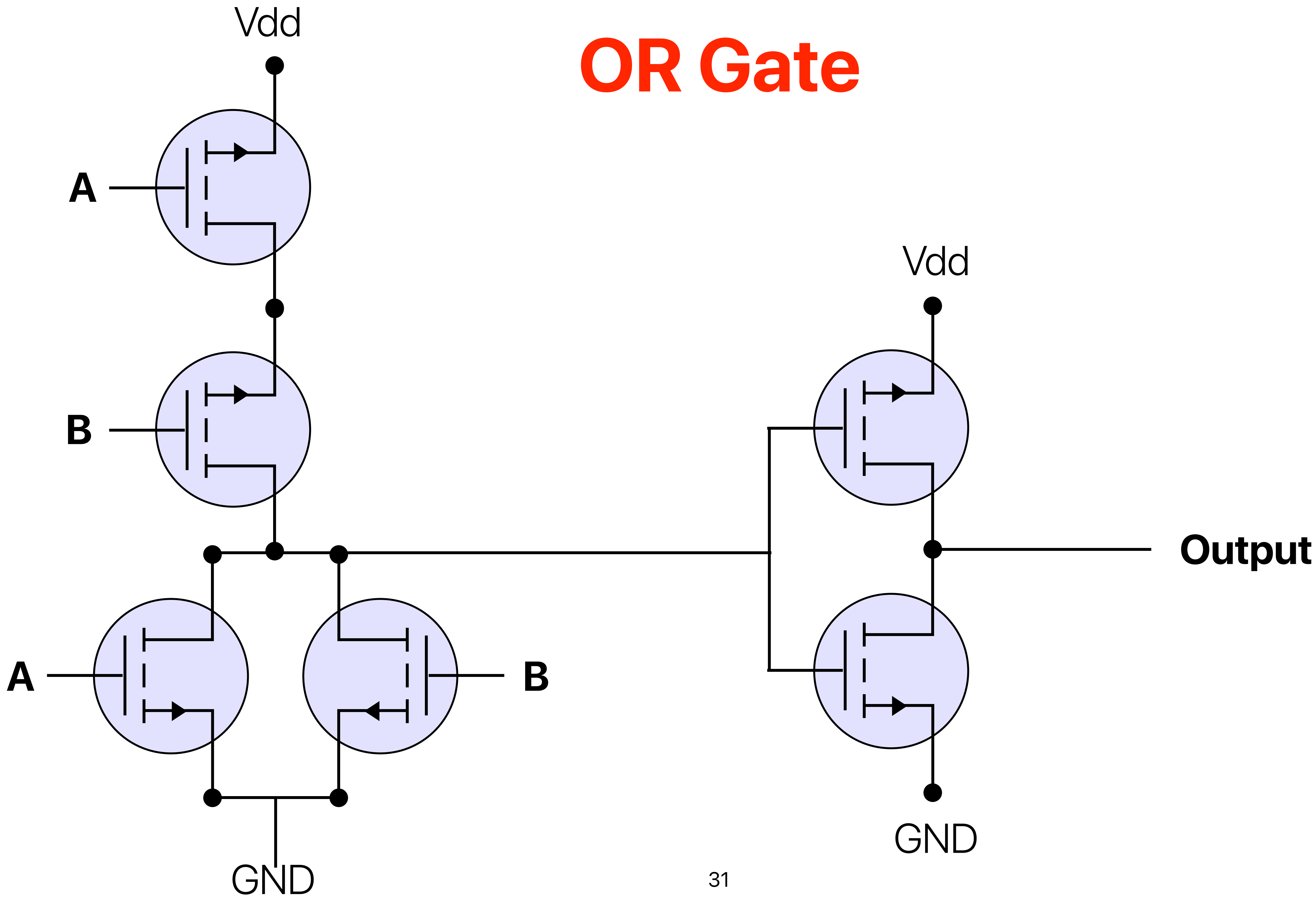
Input A	NMOS (passes 0 when on G=1)	PMOS (passes 1 when on G=0)	Output
0	OFF	ON	1
1	ON	OFF	0



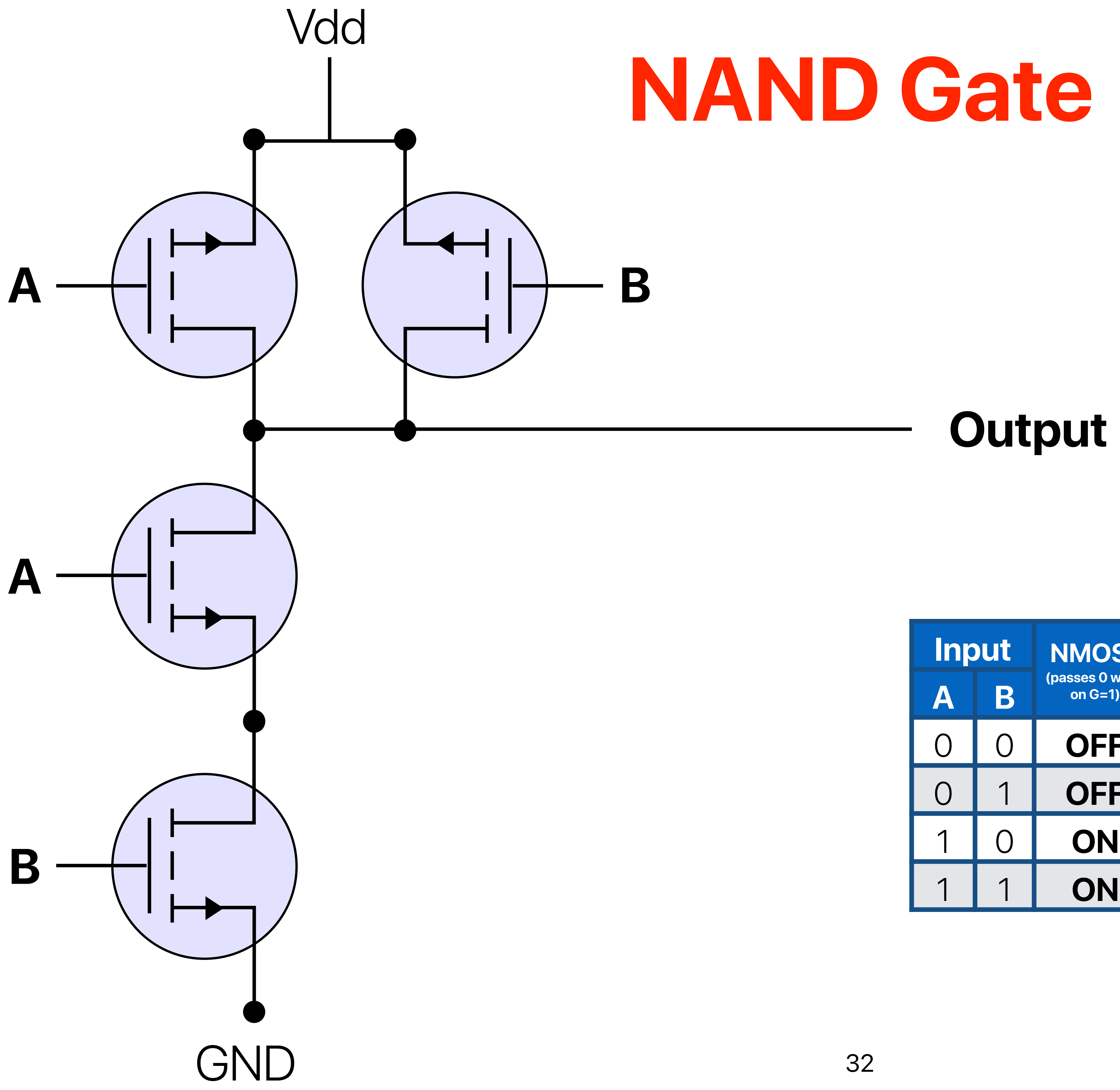
AND Gate



OR Gate



NAND Gate

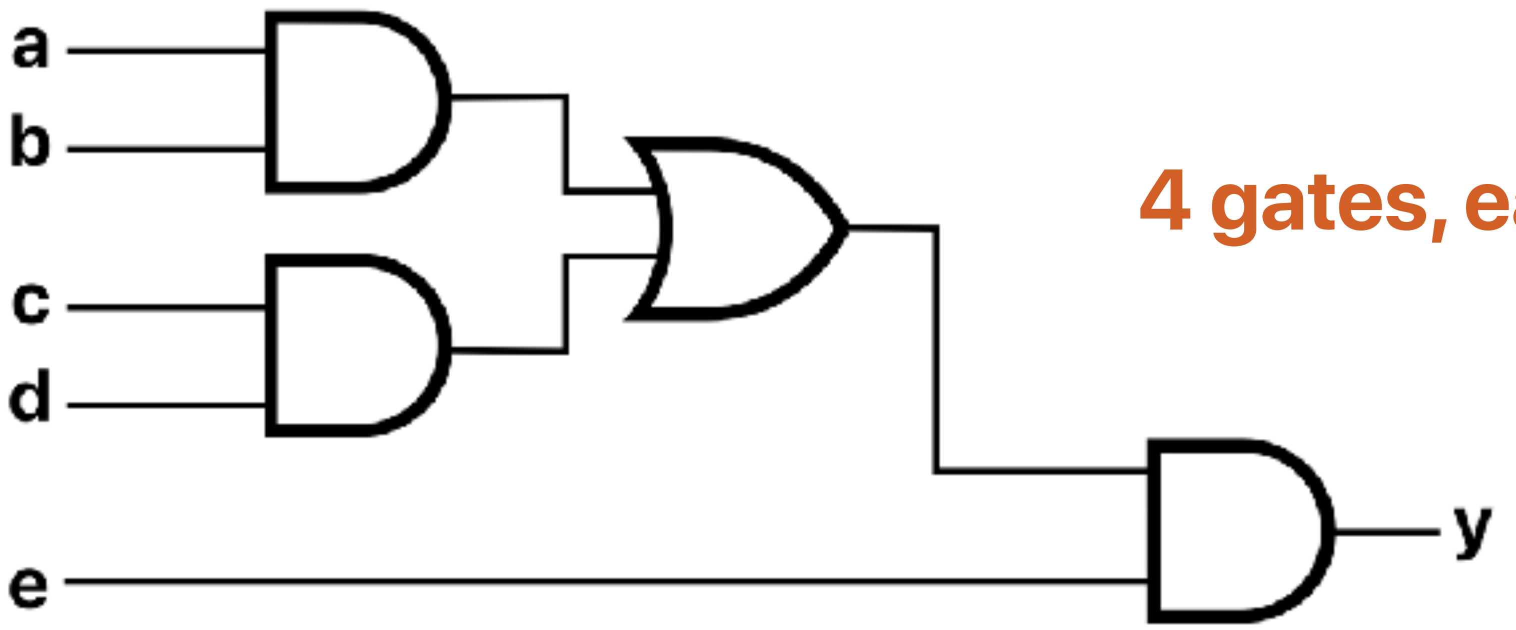


Input A	Input B	NMOS1 (passes 0 when on G=1)	PMOS1 (passes 1 when on G=0)	NMOS2 (passes 0 when on G=1)	PMOS2 (passes 1 when on G=0)	Output
0	0	OFF	ON	OFF	ON	1
0	1	OFF	ON	ON	OFF	1
1	0	ON	OFF	OFF	ON	1
1	1	ON	OFF	ON	OFF	0

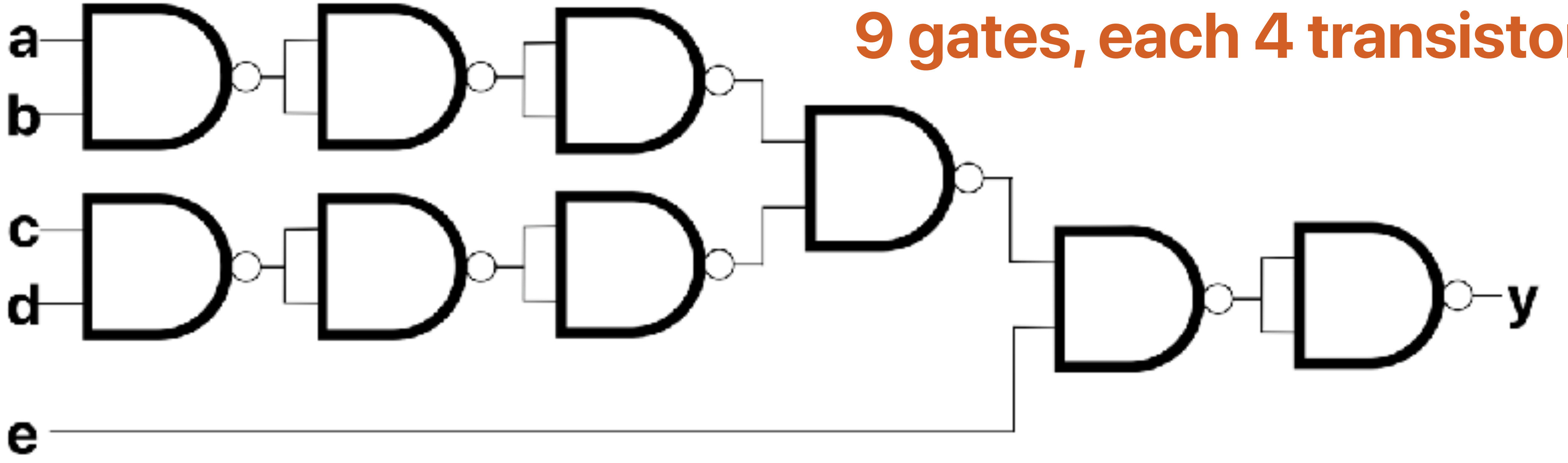
Why use NAND?

- NAND and NOR are “universal gates” — you can build any circuit with everything NAND or NOR
- Simplifies the design as you only need one type of gate
- NAND only needs 4 transistors — gate delay is smaller than OR/AND that needs 6 transistors
- NAND is slightly faster than NOR due to the physics nature

How about total number of transistors?



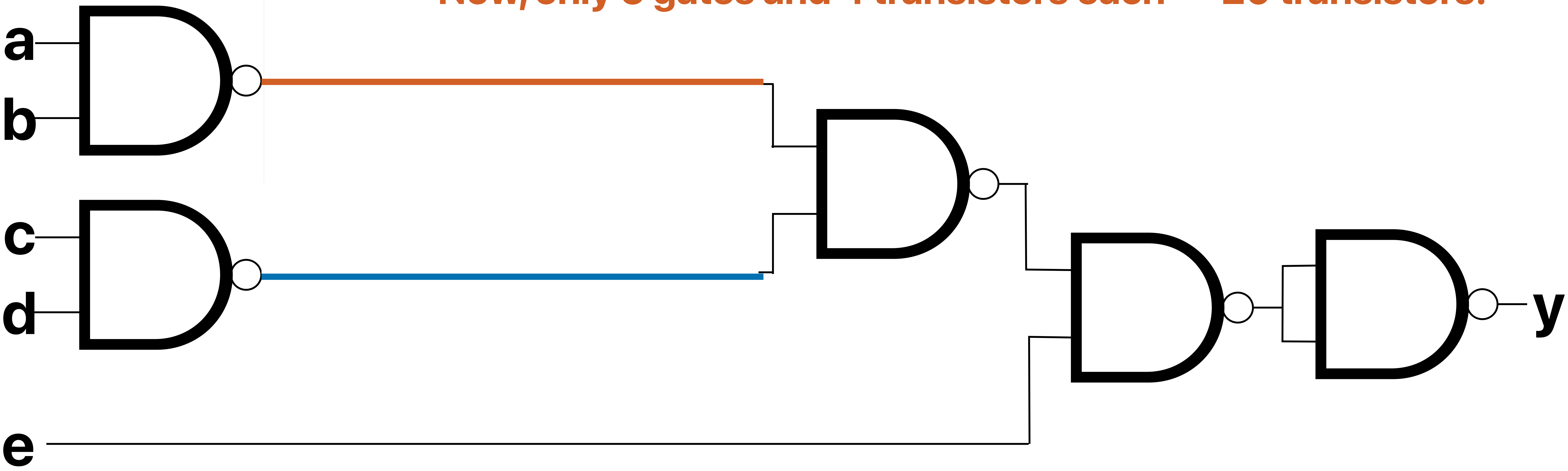
4 gates, each 6 transistors : total 24 transistors



9 gates, each 4 transistors : total 36 transistors

However ...

Now, only 5 gates and 4 transistors each — 20 transistors!



How big is the truth table of y = e(ab+cd)

- How many rows do we need to express the circuit represented by y = e(ab+cd)
  A. 5
  B. 9
  C. 25
  D. 32
  E. 64

How big is the truth table of y = e(ab+cd)

- How many rows do we need to express the circuit represented by y = e(ab+cd)
  A. 5
  B. 9
  C. 25
  D. 32
  E. 64

$$2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$$

Boolean expression is a lot more compact than a truth table!

Can We Get the Boolean Equation from a Truth Table?

Definitions of Boolean Function Expressions

- Complement: variable with a bar over it or a ′ — A′, B′, C′
- Literal: variable or its complement — A, A′, B, B′, C, C′
- Implicant: product of literals — ABC, AC, BC
- Implicate: sum of literals — (A+B+C), (A+C), (B+C)
- Minterm: AND that includes all input variables — ABC, A′BC, AB′C
- Maxterm: OR that includes all input variables — (A+B+C), (A′+B+C), (A′+B′+C)

Canonical form — Sum of "Minterms"

Input		Output
X	Y	
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

A minterm

$f(X,Y) = XY' + XY$

Sum (OR) of "product" terms

XNOR

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

$f(A,B) = A'B' + AB$

Canonical form — Product of “Maxterms”

Input		Output
X	Y	
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

A “maxterm

$f(X,Y) = (X'+Y')(X' + Y)$ ← Product of maxterms

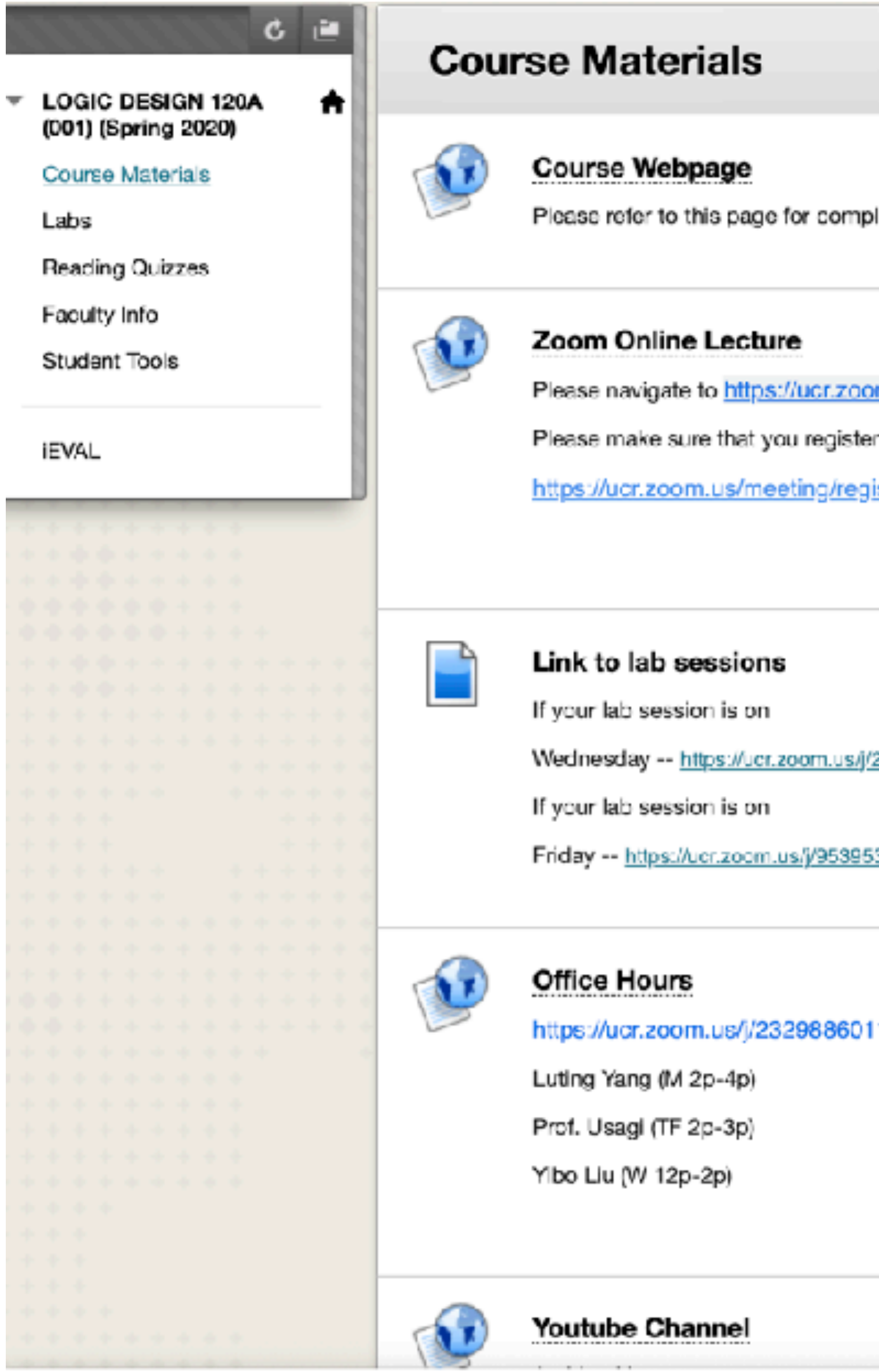
XNOR

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

$f(A,B) = (A'+B)\ (A+B')$

Announcement

- Please also register yourself to the following two
  - Please register to your corresponding lab sessions
    - The link is under iLearn > course materials
  - Please register to office hours
    - The link is also under iLearn > course materials
- Reading quiz 2 will be up tonight
  - Under iLearn > reading quizzes
- Lab 1 due 4/7
  - Submit through iLearn > Labs



Electrical
Computer Science
Engineering 120A

つづく